图文 046、动手实验：自己动手模拟出对象进入老年代的场景体验一下

1155 人次阅读 2019-08-15 07:00:00

详情 评论

动手实验：

自己动手模拟出对象进入老年代的场景体验一下（下）

给大家推荐一套质量极高的Java面试训练营课程：

作者是中华石杉，石杉老哥是我之前所在团队的 Leader ，骨灰级的技术神牛！

大家可以点击下方链接，了解更多详情，并进行试听：

21天互联网Java进阶面试训练营（分布式篇）

1、前文回顾

上篇文章给大家分析了一下对象是如何通过动态年龄判定规则进入老年代的，同时让大家自己动手去模拟写代码体验一下对象达到15岁之后自动进入老年代的场景。

今天这篇文章我们就来给大家示范一下，对象是如何在Young GC过后因为放入下Survivor区域，就直接进入老年代了。

有了之前几篇文章的铺垫，今天的文章大家理解起来就简单多了。

2、示例代码

先来看看下面的示例代码：

```
public class Demo1 {

    public static void main(String[] args) {
        byte[] array1 = new byte[2 * 1024 * 1024];
        array1 = new byte[2 * 1024 * 1024];
        array1 = new byte[2 * 1024 * 1024];

        byte[] array2 = new byte[128 * 1024];
        array2 = null;

        byte[] array3 = new byte[2 * 1024 * 1024];
    }

}
```

3、GC日志

然后我们使用之前的JVM参数来跑一下上面的程序，可以看到下面的GC日志：

0.421: [GC (Allocation Failure) 0.421: [ParNew: 7260K->573K(9216K), 0.0024098 secs] 7260K->2623K(19456K), 0.0026802 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]

Heap

par new generation total 9216K, used 2703K [0x00000000fec00000, 0x00000000ff600000, 0x00000000ff600000)

 eden space 8192K, 26% used [0x00000000fec00000, 0x00000000fee14930, 0x00000000ff400000)

 from space 1024K, 55% used [0x00000000ff500000, 0x00000000ff58f570, 0x00000000ff600000)

 to space 1024K, 0% used [0x00000000ff400000, 0x00000000ff400000, 0x00000000ff500000)

concurrent mark-sweep generation total 10240K, used 2050K [0x00000000ff600000, 0x0000000100000000, 0x0000000100000000)

Metaspace used 2782K, capacity 4486K, committed 4864K, reserved 1056768K

 class space used 300K, capacity 386K, committed 512K, reserved 1048576K

3、一步一图来分析GC日志

接着我们一点点来分析一下，首先看如下几行代码：

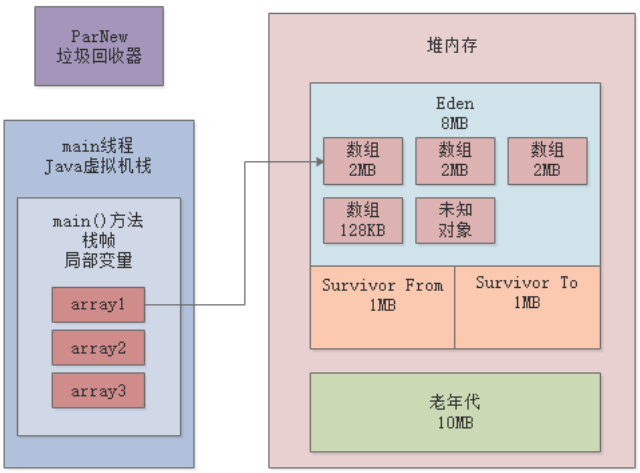
```
byte[] array1 = new byte[2 * 1024 * 1024];
array1 = new byte[2 * 1024 * 1024];
array1 = new byte[2 * 1024 * 1024];

byte[] array2 = new byte[128 * 1024];
array2 = null;
```

上面的代码中，首先分配了3个2MB的数组，然后最后让array1变量指向了第三个2MB数组

接着创建了一个128K的数组，但是确让array2指向了null，同时我们一直都知道，Eden区里会有500KB左右的未知对象

此时如下图所示：



接着会执行如下代码：byte[] array3 = new byte[2 * 1024 * 1024];。此时想要在Eden区里再创建一个2MB的数组，肯定是不行的，所以此时必然触发一次Young GC。

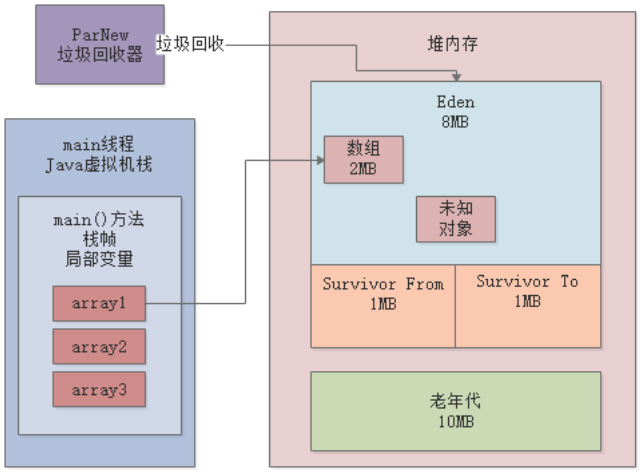
先看如下日志：ParNew: 7260K->573K(9216K), 0.0024098 secs。

这里清晰说明了，本次GC过后，年轻代里就剩下了500多KB的对象

这是为什么呢？此时明明array1变量是引用了一个2MB的数组的啊！

其实道理很简单，大家可以想一下，这次GC的时候，会回收掉上图中的2个2MB的数组和1个128KB的数组，然后留下一个2MB的数组和1个未知的500KB的对象

如下图所示。



那么此时剩下来的2MB的数组和500KB的未知对象能放入From Survivor区吗?

答案是：不能

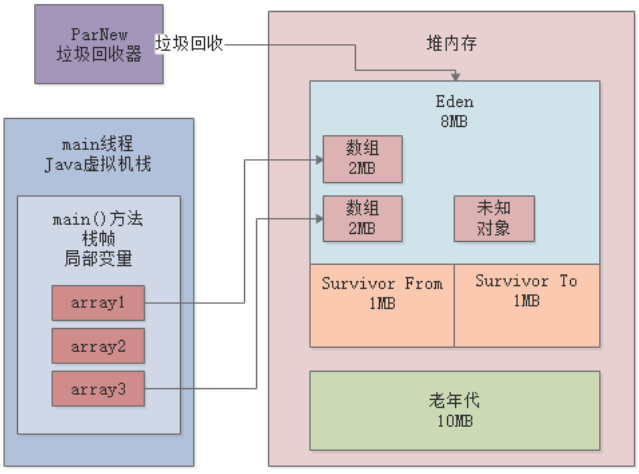
因为Survivor区仅仅只有1MB。根据我们之前说过的规则，此时是不是要把这些存活对象全部放入老年代?

答案：也不是

大家看如下日志：

eden space 8192K, 26% used [0x00000000fec00000, 0x00000000fee14930, 0x00000000ff400000)

首先Eden区内一定放入了一个新的2MB的数组，就是刚才最后想要分配的那个数组，由array3变量引用，如下图。



其次，看下面的日志：

from space 1024K，55% used [0x00000000ff500000, 0x00000000ff58f570, 0x00000000ff600000)

大家发现此时From Survivor区中有500KB的对象，其实就是那500KB的未知对象！

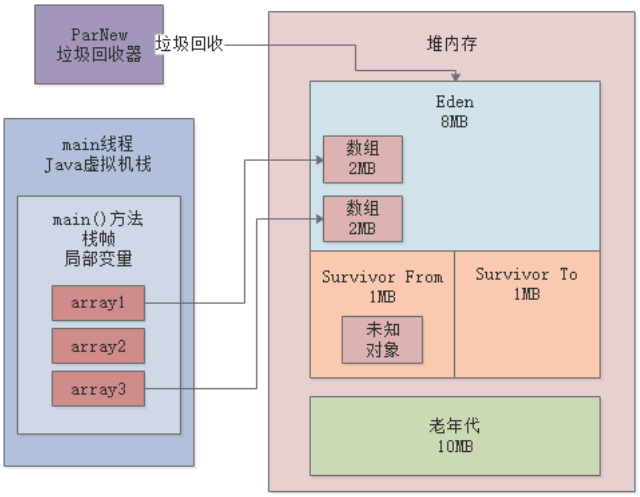
所以在这里并不是让2MB的数组和500KB的未知对象都进入老年代，而是把500KB的未知对象放入From Survivor区中！

从这里可以看到，很多细节，我们都是逐步给大家揭露开来的，之前有同学问我，如果对象放入下Survivor，是不是会有部分留在Survivor，部分进入老年代？

我当时没做明确的回答，而且之前的案例里从没提过这个细节，那是因为时机不到，当时大家不需要理解到这么细的程度。

但是现在结合GC日志，大家可以清晰的看到，在这种情况下，是会把部分对象放入Survivor区的。

此时如下图所示。

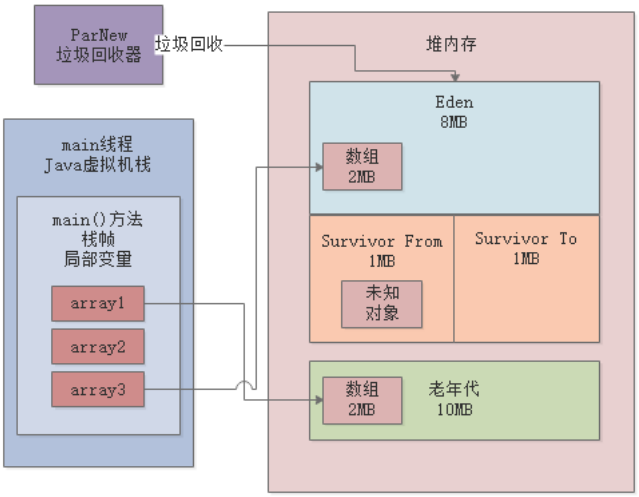


接着我们看如下日志：

concurrent mark-sweep generation total 10240K, used 2050K [0x00000000ff600000, 0x0000000100000000, 0x0000000100000000)

此时老年代里确有2MB的数组，因此可以认为，Young GC过后，发现存活下来的对象有2MB的数组和500KB的未知对象。

此时把500KB的未知对象放入Survivor中，然后2MB的数组直接放入老年代，如下图。



4、本文总结

本文篇幅不长，但是很好的给大家结合GC日志分析了Young GC过后存活对象放不下Survivor区域，从而部分对象会进入老年代的示例

在这里大家也明白了一个细节，在这种场景下，有部分对象会留在Survivor中，有部分对象会进入老年代的。

5、今日思考题

今天给大家留一个思考题，希望大家去写代码模拟一下，分配一个大对象，然后让大对象直接进入老年代，看看GC日志，是否大对象会直接进入老年代?

End